LIVEABLE CITY DIGITAL TWIN: A PILOT PROJECT FOR THE CITY OF LIVERPOOL (NSW, AUSTRALIA)

A. A. Diakite[1,*], L. Ng[2], J. Barton[1], M. Rigby[3], K. Williams[2], S. Barr[3], S. Zlatanova[1]

[1] UNSW, School of Built Environment, Sydney, New South Wales, Australia - (a.diakite, s.zlatanova, jack.barton)@unsw.edu.au
[2] Frontier SI, Melbourne, Victoria, Australia - (lng, kwilliams)@frontiersi.com.au
[3] AURIN, University of Melbourne, Parkville, Victoria, Australia - (rigby, stuart.barr)@unimelb.edu.au

Commission IV, WG IV/9

KEY WORDS: Smart city, Liveability, 3D models, CityGML, 3DCityDB

ABSTRACT:

In recent years, the concept of Digital Twin (DT) for cities is increasingly at the core of most smart city initiatives, as it has been identified as a critical tool for tackling the challenges of this century. A robust city modelling framework is essential if local, state and national governments are to move towards sustainable built environments and work together across complex multi-sectoral problems to drive impacts that improve urban liveability and climate adaptability. Furthermore, the level of collaboration and interoperability required to address these cannot be achieved without proper standardisation of DT components. The aim of this project is to develop a demonstration DT that integrates existing data using a standardised 3D format based on CityGML and that embeds analytics, such as sun exposure and tree coverage, to assess liveability within a 3D city modelling framework. Common urban features such as buildings, roads, railways, vegetation and water bodies are also processed and incorporated. Additionally, IoT sensors are integrated into the model and all processes are performed using open-source tools to improve accessibility and repeatability. Details of the workflow, including the storage of the city features in a 3D City Database (3DCityDB), the 3D upgrading of urban features commonly available as 2D data as well as a few use cases are illustrated and discussed in this paper.

1. INTRODUCTION

While the DT concept is commonly seen to have its roots in the manufacturing industry (Grieves, 2014), it is becoming increasingly valuable in a range of different domains. Typically a DT is built for a specific purpose (e.g. analysis that can help understanding and predicting) and different purposes require different data. A DT is generally agreed to be a virtual representation connected to its physical counterpart through established connection mechanisms (Tao et al., 2018) that, some would argue, need to be real-time enabled. DTs of cities have the potential to transform the design, management and performance of the built environment. They are expected to play a critical role in tackling the big challenges that cities are facing by enabling smarter urban planning, energy management, transportation, liveability, and many other aspects (Ivanov et al., 2020).

Interoperability between different systems has improved to the point where it has become technically feasible to assemble a set of technologies that combine to form near-real-time representations of the world. In the field of Geographic Information Sciences (GIS), the concept of 3D city modelling that has been trending for over two decades (Förstner, 1999, Kolbe et al., 2005, Emgård et al 2007, Billen et al 2014, Biljecki et al., 2015) fits well to the DT concept. How-ever, just as for the Architecture, Engineering and Construction (AEC) sector and its Building Information Modelling (BIM) paradigm (Boje et al., 2020), advanced geometric and semantic representations are good for virtualisation of physical assets but not enough to be categorized as a DT (Wright and Davidson, 2020). It is therefore natural for DTs of cities to adopt 3D city models as their virtual representations, while other necessary components such as the 3D analysis and real-time information integration are still being investigated (Francisco et al., 2020).

Many DT initiatives have been documented in the recent literature. The Helsinki 3D (Finland) project is discussed in (Ruohomäki et al., 2018) with an emphasis on the history and the different components that led to its current status from its earliest 3D model initiative in the 1980's until nowadays. Other critical aspects such as open data and privacy issues are also mentioned. The DT of the city of Zurich (Switzerland) and its different components are introduced in (Schrotter and Hürzeler, 2020). The emphasis is put on the openness of the data used to build the DT and the different applications implemented in order to support efficient urban planning. Another public and open DT is the Docklands area in Dublin (Ireland), discussed in (White et al., 2021), with most of the focus put on the possibility for the citizens to interact with the DT and bring their feedback to urban planning projects. More recently, the first steps towards a DT of Sofia city (Bulgaria) is introduced in (Dimitrov and Petrova-Antonova, 2021). The paper mainly discusses the creation of the 3D model of the city with a framework that leverage existing data. All of these previously cited DTs, along with many other renowned ones (e.g. Virtual Singapore, i-URBAN in Japan, etc.) (Gobeawan et al., 2018, Akahoshi et al., 2020) have in common the use of an international 3D city model standard, namely CityGML (Kolbe et al., 2005). With its recently released third version (Kutzner et al., 2020), it is the most advanced standard available for the virtual representation component of DT of cities. In fact, the level of collaboration and interoperability expected from a DT cannot be reached without a proper standardisation of DT components.

This paper presents a project exploring the process of building a full-stack DT using primarily open-source components. The processes covered by the project includes ingesting existing three-dimensional (3D) geospatial data into a database (3DCityDB/PostGIS), connecting to Internet-of-Things (IoT)

* Corresponding author

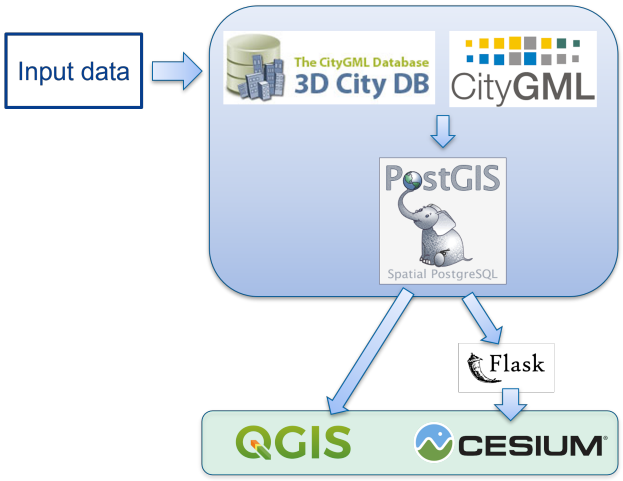


Figure 1. Workflow adopted for the implementation of WP1.

Data set	Geometry type	Format
Buildings	MultiPolygonZ	GDB
Roads	MultiLineStringZ MultiPolygon	GDB & SHP
Railways	MultiLineStringZ	GDB
Water bodies	MultiLineStringZ MultiPolygon	GDB & SHP
Vegetation	MultiPolygon PointZ	GDB & SHP
Terrain (DEM)	Raster (grid)	GeoTIFF
IoT Sensors	Point	GeoJSON

Table 1. List of data sets collected and used in the project.

Application Programming Interface (API) services, enabling semantic mark-up/update and deploying to both standard GIS desktop software for further processing/analysis (QGIS) and also visualisation in 3D web-based interactive environments (CesiumJS). The main contribution is the demonstration that a functioning digital twin can be built using already existing data and open-source technologies. For this purpose, the Liverpool precinct in Western Sydney was chosen as a case-study area, and three use-cases identified: (1) 3D shadowing analysis for events planning, (2) canopy planning based on shadowing from existing trees and (3) real time visualisation of sensor feeds.

2. WORKFLOW AND DATA SETS

The project comprises 3 main parts organised as working packages (WP):

- WP1: Building of DT (based on standards);
- WP2: Use of DT;
- WP3: Update of DT.

WP1 involves the collection and integration of the input data sets in a standardised way, their storage and access management, as well as their visualization in several interfaces. For WP2, use cases related to thermal comfort has been selected for implementation on the created DT, including the shadow computation and analysis with respect to available sensor and vegetation data. WP3 is dedicated to the investigation and implementation of proper mechanisms for the update of the DT data from the front-end (this is current work in progress that will not be included in this publication).

Figure 1 illustrates the workflow that has been adopted for implementing WP1, which is the basis of the whole project. At the core of the workflow, the 3D City Database (3DCityDB) was chosen as a geo database to store, represent, and manage the data based on a standard spatial relational database (Yao et al., 2018). Its database schema implements the CityGML standard with semantically rich and multi-scale urban objects. Its implementation is based on PostgreSQL/PostGIS (Hsu and Obe, 2021) giving thereby access to powerful queries and analysis tools directly on the stored data. It is also well supported and documented, and offers convenient tools to import and export data in different formats. In terms of interfaces for visualisation of the data, QGIS (QGIS Development Team, 2022) and Cesium (Cesium GS, 2022) were chosen. The choice of QGIS is motivated by its direct compatibility with Postgres/PostGIS data along with its wide range of powerful tools allowing manipulation and analysis. However the 3D visualisation capabilities of QGIS are still very limited, which is the reason why Cesium is also used for more advanced visualisation and front-end interactions. Because Cesium cannot directly communicate with the database, we established an API between PostgreSQL and Cesium using the python library Flask.

At the beginning of the project, we were provided with many spatial data sets of the test area. They all have different data structure, objects and descriptions. Therefore, a critical task for the integration and storage of the data consisted in mapping their classes and attributes to those of CityGML for importing them in 3DCityDB. Focus was put on the data listed in Table 1 that represent the main features of a 3D city model. The data sets are a mix of 2D (geometries of type MultiPolygon or Point) and 3D (geometries of type MultiPolygonZ, MultiLineStringZ or PointZ) vector data and raster for the Digital Elevation Model (DEM). They are representative of common data that local councils, municipalities and governmental spatial services would possess.

However, not all these data sets are directly suitable for our CityGML based 3D DT. Further processing is required to upgrade some of them into 3D data and map them suitably to 3DCityDB. For this purpose, we developed custom Python and SQL scripts to complete the processing, including the generation of a 3D Triangle Irregular Network (TIN) of the terrain taking into account all the other city features sitting on it. We discuss the mapping of the data sets to classes of 3DCityDB in the following section.

3. TRANSFORMATION AND STORAGE OF THE DATA INTO 3DCITYDB

3.1 Buildings

This data set describes 3D geometries of building throughout the city of Liverpool. They are comparable to the Level of Detail (LoD) 2 of the CityGML standard. Features of the data set

contain 51 attributes in total, plus their geometric information. 3D Buildings are a direct fit to the building model of CityGML, which is in turn characterised by 34 attributes (including geometry). However, direct matching between attributes of the data set and CityGML is pretty limited as only 3 CityGML attributes were identified as having a direct match (if we ignore mapping of the IDs):

- *roofType*
- *measuredHeight*
- *lod2MultiSurface*

This small number of matches should be contrasted with the fact that about 20 CityGML building attributes are dedicated to different LoD representations while our data has only one of them. Furthermore, all attributes without direct match can still be stored within the standard via the *_genericAttributes* modules.

Another limitation of the building data set is its lack of semantic information related to the building components. Such missing information would result in a poor CityGML model, as one of the main strengths of that standard lies on its support of semantic data. Because it is an LoD2 model, the faces of the buildings need to be classified in 3 main CityGML semantic classes: *RoofSurface, GroundSurface* and *WallSurface*. We perform the classification based on a face orientation approach: vertical faces are considered as walls; the rest are considered as roofs if they lie above the centre of gravity of the building, and ground otherwise. This gives us a richer data set where component such as roofs can be specifically queried, which could be useful for applications such as solar exposure estimation on roofs.

Finally, the processed data is stored in the corresponding tables of our 3DCityDB instance in PostgreSQL. For the building data set, 5 tables of 3DCityDB needed to be filled: *CITYOBJECT, BUILDING, CITYOBJECT_GENERICATTRIB, THEMATIC_SURFACE* and, *SURFACE_GEOMETRY*. The *CITYOBJECT* table is the main table of 3DCityDB, and every feature of the model needs to be registered in it. For the rest, the building table is the one specific to the building classes and that will take the matched attributes (id, *roofType* and *measuredHeight*) but not the geometry. The latter goes to the *SURFACE_GEOMETRY* table where each face is stored as a separate entry (polygon) along with its specificities. The *THEMATIC_SURFACE* is the table allowing to make the link between the classified surfaces and their corresponding buildings. Finally, the *CITYOBJECT_GENERICATTRIB* is the table where all the attributes that did not find a direct match go, while maintaining a reference to the building that they belong to (through *building_id*). A few more tables would have been considered if our input data had texture information (e.g. *SURFACE_DATA*, *APPEARANCE*, etc.).

3.2 Roads and Railways

Both the roads and railways datasets correspond to the transportation model of CityGML/3DCityDB. In the standard, the transportation schema is defined by a superclass *_TransformationObject* which can aggregate three other classes:

- *TransportationComplex*,
- *TrafficArea*, and
- *AuxiliaryTrafficArea*.

3DcityDB table	Data set
BUILDING	Buildings
CITYOBJECT	All data
CITYOBJECT_GENERICATTRIB	All data
GENERIC_CITYOBJECT	IoT Sensors
SURFACE_GEOMETRY	Buildings
THEMATIC_SURFACE	Buildings
TRANSPORTATION_COMPLEX	Railways Roads
WATERBODY	Water bodies

Table 2. List of 3DCityDB tables altered based on corresponding data sets.

TransportationComplex is the main class to represent roads, tracks, railways, squares, etc. It is composed of the parts *TrafficArea* and *AuxiliaryTrafficArea*. In our case, the initial 3D road and railway features are represented by line geometries (MultiLineStringZ), which means they do not provide enough details to be classified into *TrafficArea* and *AuxiliaryTrafficArea*. The other road data that we have is 2D (MultiPolygon), and needs to be upgraded to 3D. This, as well as the upgrade of the railways, is discussed in Section 3.5 as the upgrading process relies on the 3D terrain. Consequently, we only consider the class TransportationComplex and its attributes at this stage. *TransportationComplex* has 10 attributes (13 in 3DCityDB) while both our 3D road and railway data sets contain 35 attributes in total. Here again, apart from the geometry and potentially the IDs no direct matching can be identified between the attributes. Thus, we used the generic attributes to not lose any information.

Three tables are altered: *CITYOBJECT, TRANSPORTATION_COMPLEX* and, *CITYOBJECT_GENERICATTRIB*. The *CITYOBJECT* table takes entries for the same reason as explained in Section 3.1. Value entries to the *TRANSPORTATION_COMPLEX* table are dedicated to the columns id, *objectclass_id* (which is 44 for roads and 45 for railways) and *lod0_network* for the geometry data. Additional consideration was taken with respect to the road dataset and its attributes such as *functionhierarchy*, *classsubtype*, etc. which allows us to distinguish between the types of road (e.g. pathway, local road, etc.). We could thereby categorise pathways as tracks in CityGML/3DCityDB (*objectclass_id* = 43). The complete list of values for the *objectclass_id* attribute is provided in the documentation of 3DCityDB. All the other attributes that did not find a direct match are stored in the *CITYOBJECT_GENERICATTRIB* table.

3.3 Water bodies

Another input data set describes the water bodies traversing the city. Such kind of data corresponds best to the Waterbody class of CityGML. Water bodies can be described with a lot of detail thanks to other classes for surface information, such as WaterSurface, WaterClosureSurface or WaterGroundSurface. However, similarly to the transportation data sets, the water bodies

data set only describe basic geometries (lines) of the water bodies in question. Therefore, the class Waterbody is the only one that can handle them as it provides possibilities to store MultiCurve geometries. In terms of attribute, the scenario is similar to the transportation objects, with 12 attributes of WaterBody that do not directly match with the 23 attributes of our dataset, except for the geometry.

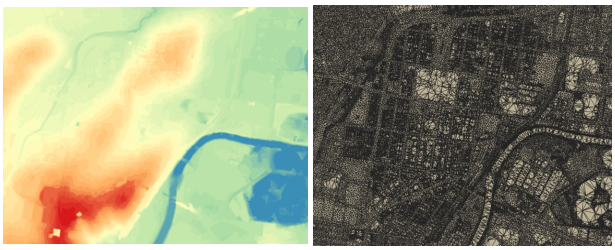
Here again, no further processing or enrichment of the data is necessary. The information is inserted in the database as is in three following tables: *CITYOBJECT, WATERBODY* and, *CITYOBJECT_GENERICATTRIB*. Every water entity is registered in the *CITYOBJECT* table, like any other features of the model. Then the id, *objectclass_id* and *lod0_multi_curve* columns of the *WATERBODY* table are updated accordingly. Again, all the attributes that did not find a direct match are stored in the *CITYOBJECT_GENERICATTRIB* table.

3.4 Vegetation and IoT

The vegetation data set is also a basic one providing only tree locations in 3D (PointZ, with the Z coordinate corresponding to the elevation of the tree), and their height. The data was produced from classified LiDAR and the SolitaryVegetationObject class of the Vegetation model of CityGML is its best fit because it represents individual trees rather than coverage. Regarding the geometry, the SolitaryVegetationObject class is associated with a geometry class representing an arbitrary GML geometry or an implicit geometry, and may have a different geometry in each LoD. Also, height is one of its default attributes, therefore our data populates the following two 3DCityDB tables: *CITYOBJECT* and, *SOLITARY_VEGETAT_OBJECT*. The trees are registered in the *CITYOBJECT* table with an *objectclass_id*. The geometric data is stored in the *lod1_other_geom* column and the height in the column of the same name.

A wide range of IoT sensors is already deployed and functional in the test site of the project, covering different themes (e.g. environment, transport, etc.) and accessible through an open data portal API. While the sensed data themselves do not need to be explicitly stored in our DT database, unless necessary (e.g., very resource consuming analysis), only the sensors (physical location) and their metadata need to be integrated in our city model. This enables management of explicit information about the sensor devices, their properties and spatial location, which allows efficient spatial analysis with the sensed data. Unfortunately, there is no class that provides direct compatibility with IoT sensors as features in CityGML 2.0, which is used in our project. This is expected to be possible in the newly released version 3.0 where they might be considered as City Furniture instances, but we have not investigated this further. Meanwhile, one workaround that we adopted is to use GenericCityObject class (generic city object concept). Similar to the generic attributes, this concept allows for the storage and exchange of 3D objects which are not covered by any explicitly modelled thematic class within CityGML 2.0 or which require attributes not represented in CityGML. In 3DCityDB, the sensors' information is inserted in the three following tables: *CITYOBJECT, GENERIC_CITYOBJECT* and, *CITYOBJECT_GENERICATTRIB*.

Every sensor device gets registered in the *CITYOBJECT* table and is provided with a unique *cityobject_id*. The same *cityobject_id* is used as id in the *GENERIC_CITYOBJECT* table, where an *objectclass_id* ($= 5$) is also recorded and where a Point geometry is in the *lod0_other_geom* columns using the longitude and latitude values of the sensor. Because the height values of the sensors are missing in the datasets, an altitude of 0 is temporarily used as the Z coordinate of the points. Later on, we are aiming to update this information as it will be relevant for the planned spatial analysis. Finally, all the other metadata of the sensors are stored in the *CITYOBJECT_GENERICATTRIB* table.



(a) (b)

Figure 2. (a) Coloured DEM (blue is the lowest elevation and red the highest). (b) Derived raw TIN.

3.5 Terrain

The original terrain data is a DEM of area of interest, which is in the form of a 1m $\times$ 1m raster GRID, obtained from the Elvis platform of the Foundation Spatial Data (FSDF - Elevation and Depth). However, such raster data is not suitable for a polygon-based approaches such as the 3D shadow estimation discussed in Section 5. Therefore, we generated a 3D TIN directly from the raw DEM in an approach similar to (Yan et al., 2019b).

The TIN in Figure 2(b) considered the footprints of the buildings only. The other features that can be distinguished are due to their contrast in elevation with the rest of the terrain (e.g. between the ground and the water bodies), but they were not explicitly included. Despite the higher resolution, this TIN that contains approximately 5.8 million triangles is not efficient to work with and most applications do not require that much detail. We therefore decided to generate a simpler TIN that would minimise the number of polygons while including all the features of importance (buildings, roads, railways, and water bodies). As previously mentioned, all these features are available and already stored in the database, however, apart from the building data, all of them need to be upgraded to 3D to fit to the purpose. First, an integrated TIN is created based on the 3D boundaries of the other features (meaning MultiLineStringZ is created from their 2D polygonal shapes), and then their full 3D polygonal shapes (MultiPolygonZ) are recovered from the generated TIN. All the processes described here were performed with QGIS and PostGIS.

3.5.1 Roads, railways and water bodies 3D boundaries generation Besides the 3D lines with attributes discussed in Section 3.2, the 2D road polygons extracted from a Pavement Management System (PMS) were provided to us by the local council of the study area. The data describes complex shapes excluding the pavements on the sides and in-between lanes. It was preferred to another data set describing road cadastre because it allows the discrimination between pedestrian and vehicle spaces. Figure 3(a) illustrates the road polygons in their 2D states and their boundary vertices (in red). All the boundaries were converted to 3D by a DEM-based sampling of their vertices to recover their elevation (z coordinate).

Similarly to the road data set, the railways originally come in the form of 3D lines. However, unlike the roads, it is not ne-

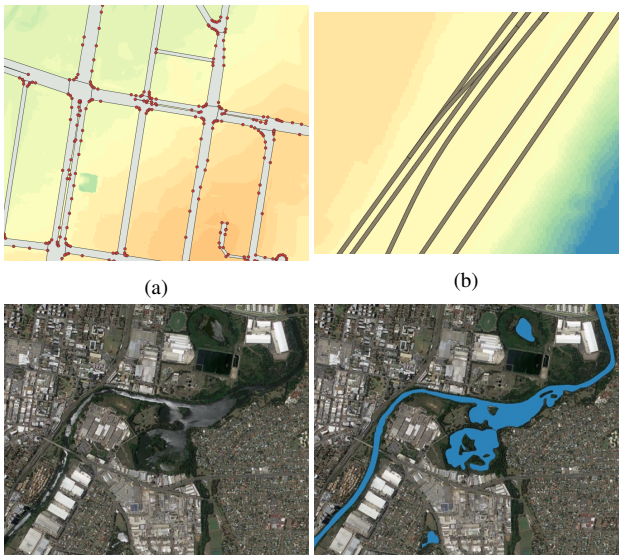


Figure 3. (a) Road polygons (grey) and their boundary vertices (red) sampled against the DEM. (b) Railway polygons (dark grey) obtained by buffering their lines to the local standard width (1.435m). (c) Satellite image of some water bodies and (d) their corresponding polygons.

cessary to seek a polygonal version as this can be fairly derived from the line, assuming a regular width for the feature. We therefore produced polygons using the buffering tool of QGIS with a standard width of a 1.435m (4 ft 81/2 in), as specified by the Australian Bureau of Infrastructure and Transport Research Economics (BITRE, 2022) (Fig. 3(b)).

3D lines of water bodies cannot just be dilated with a buffer like the railways. Therefore, we looked for other data sources and found an accurate hydrology polygon dataset of the region provided by GeoScience Australia, via the AURIN Portal (Sinnott et al., 2015) (Fig. 3(c) and (d)). But unlike the 3D lines data, creeks and other temporary water bodies were not represented in the data. We therefore decided to exclude the creeks and dealt only with parts described as polygons. The same process of elevation sampling from the DEM is performed and the 3D boundaries of the inner and outer rings were extracted.

3.5.2 Generation of the Integrated TIN With all the features ready (building footprints, roads, railways, and water bodies), we proceeded to the generation of a TIN that integrates them all. We used the 3D constrained Delaunay triangulation implementation of PostGIS. For this, we provided as input the 3D boundary lines of the features as constraints for the triangulation. The resulting TIN which can be seen in Figure 4 counts 28K polygons. It is more densely triangulated in areas with many features, making it suitable for an efficient use in spatial analysis while preserving a good elevation information for all the features. Since we have the information of all the inputs, we could augment the final TIN by associating a label attribute to each triangle based on its corresponding feature. Based on this information, the full 3D polygonal shapes of all the roads, railways and water bodies are finally reconstructed and their corresponding database classes augmented accordingly.

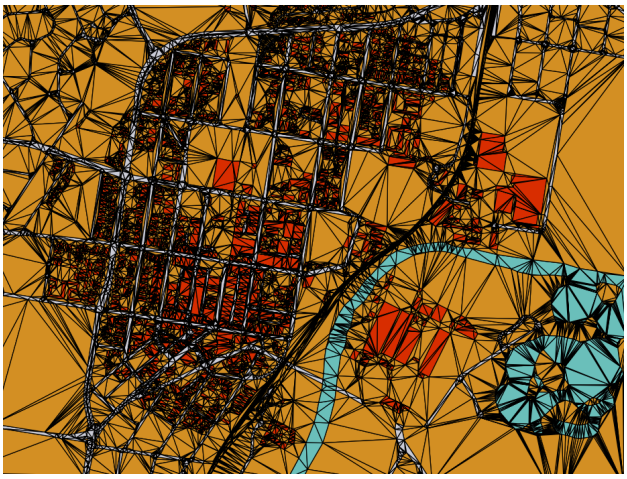


Figure 4. Final TIN integrating all the features: building footprints (red), ground (brown), roads (light grey), railways (dark grey) and water bodies (blue).

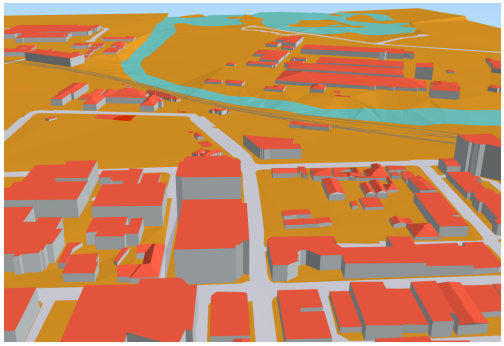


Figure 5. All the data sets visualised in QGIS (with the Qgis2threejs plugin).

4. VISUALISATION IN QGIS AND CESIUMJS

4.1 QGIS

The format adopted for our DT storage (3DCityDB/PostGIS) enables the direct access and visualization of the data in QGIS, from which the database can be directly connected to and the geometry directly queried and visualized, as illustrated in Figure 5. The native 3D capabilities of QGIS are still limited though, thus we used the Qgis2threejs plugin. It offers smoother 3D visualisation but the memory handling is still a challenge for large scale models.

4.2 CesiumJS

Another platform for visualising DT data on the web is Cesium (Cesium GS, 2022). Cesium supports a wide range of spatial data (GIS, BIM, photogrammetry, etc.) in various formats, including CityGML. 3D contents are all converted to 3D Tiles, an open specification format that optimises the streaming of large 3D data through the Cesium viewer. To access to such format, one must use the dedicated web app by creating a Cesium ion account (which is the proprietary version of Cesium, not to confuse with CesiumJS, the open-source version) and uploading the data for them to be converted to 3D Tiles and become accessible through a RestAPI. Once the tiles are created, they can be streamed to CesiumJS. Besides 3D Tiles, CesiumJS supports several other formats (KML, GeoJSON, OBJ, etc.).

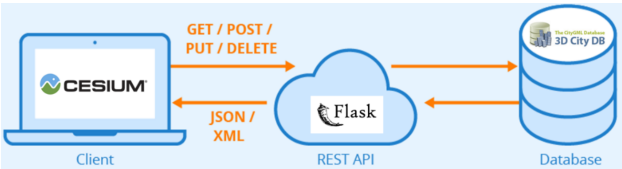


Figure 6. Diagram of the API requests established between Cesium and 3DCityDB through Flask.

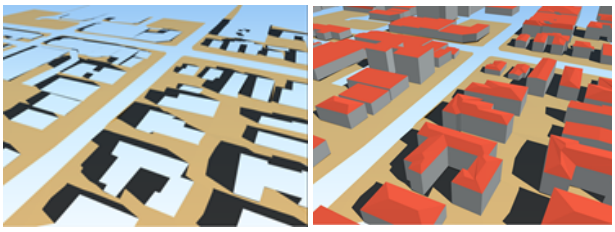
Figure 7. Data queried from 3DCityDB through a custom API and fed to CesiumJS as GeoJSON. Attributes of the features are also queried along their geometries, as shown by the description of a selected sensor on the scene.

However, all these approaches imply file-based processes between the data custodians and Cesium Ion. This is not convenient for dynamic databases where information is often updated, as one would expect for the database of a city's DT. Since there is no official support of Web Feature Service (WFS) for Cesium, we opted for implementing a custom API for accessing 3DCityDB data. As a web interface based on JavaScript, CesiumJS can send and receive HTTP requests to servers. Therefore, with the support of GeoJSON files, an ad-hoc workaround to the file issue is to send to CesiumJS results of database queries formatted as JSON through an API. We adopted this approach and used Flask, a Python library to set up a REST API. Flask communicates with 3DCityDB via SQL queries (using the psycopg Python package). Figure 5 illustrates the communication established between the client (CesiumJS) and server side containing the database (3DCityDB) and the API (Flask).

The queries sent to 3DCityDB through Flask should return valid GeoJSON data to be readable by Cesium. To produce such outputs, we rely on PostGIS' *ST_AsGeoJSON* function that formats the results of the queries. A useful new feature of PostGIS (from v3.0.0) allows to create GeoJSON from a full database record, meaning a row containing geometry and other attributes. From the Cesium side, a simple call to the API request needs to be performed to obtain GeoJSON data. The latter is then injected in the viewer resulting in visualisations as illustrated in Fig. 7.

5. USE CASES

With WP1 completed (see Section 2), we can use the features of the built DT database to perform some analysis. Three related use cases have been explored: (1) a 3D shadowing analysis for events planning, (2) canopy planning based on shadowing and existing trees and (3) real time visualisation of sensor feeds.



(a) (b)

Figure 8. (a) Shadowed and sun-exposed ground polygons. (b) Building features causing the shadows.

The goal of the first use case is to perform shadow analysis on selected locations to determine their potential sunlight exposure during a planned event and see how this can be mitigated (e.g., by placing temporary shade). To achieve this, a shadow computation algorithm was developed and the CesiumJS user interface (UI) modified to allow a user to pick a specific location for an event. Additionally, the user has to pick a start and end date to run the analysis; for every hour within that range, we compute three outputs:

- the non-ground shadowed polygons,
- the shadowed polygons on the ground, and
- the sun-exposed polygons on the ground.

This distinction in different types of outputs is made to facilitate further analysis relying on them. Furthermore, the outputs are currently saved as GeoJSON files to allow their storage, sharing and import in common GIS tools such as QGIS, as shown in Fig. 8. The categorisation of the TIN based on the labels from the features helped improve the shadow analysis by allowing focus on specific features. Thereby, the ground polygons produced specifically correspond to areas accessible to pedestrians. It remains possible to control this and include other features like roads for example, when necessary. Based on the same generated data, but with additional functionalities, this use case can be extended with pedestrian routing that allows the use of the shadowed parts of the streets. Advanced indoor/outdoor navigation as described in (Yan et al., 2019a), could also be envisaged but that would require adding models with indoor details (e.g. BIM) to the DT.

For the second use case, we checked the unshaded areas against the canopy/vegetation data available to us. Unfortunately, the data is just a point layer describing tree locations and with their height and elevation. Since this is not enough to guess the amount of shadow that they cast, we applied a heat map of the trees assuming an approximate area around the trees that we consider as shadowed (see Fig. 9). It is visible that the studied area is under-covered in terms of trees and would benefit from more trees that would bring more shadow. Although most of the exposed zones are open parking areas and private gardens, the main streets are still not covered enough. The block at the bottom right side of the image reflects an area where no 3D building data were available.

Finally, we developed a visualisation interface for the IoT sensors available in area of study as a third use case. While the data and some dashboards can be found on the city's open data portal, visualising the feed on a 3D map gives a better spatial context to the information. Figure 10 shows the feed

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W2-2022-45-2022 | © Author(s) 2022. CC BY 4.0 License.

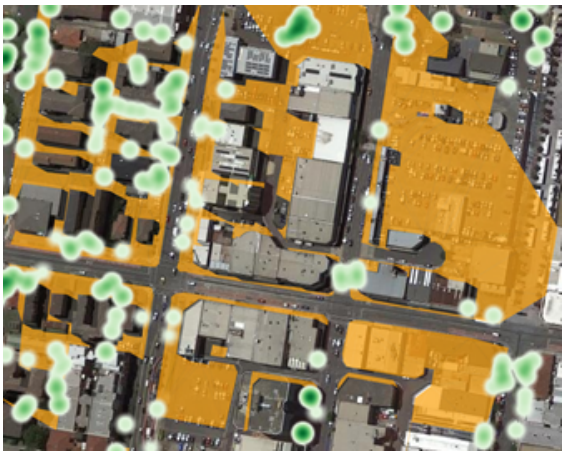


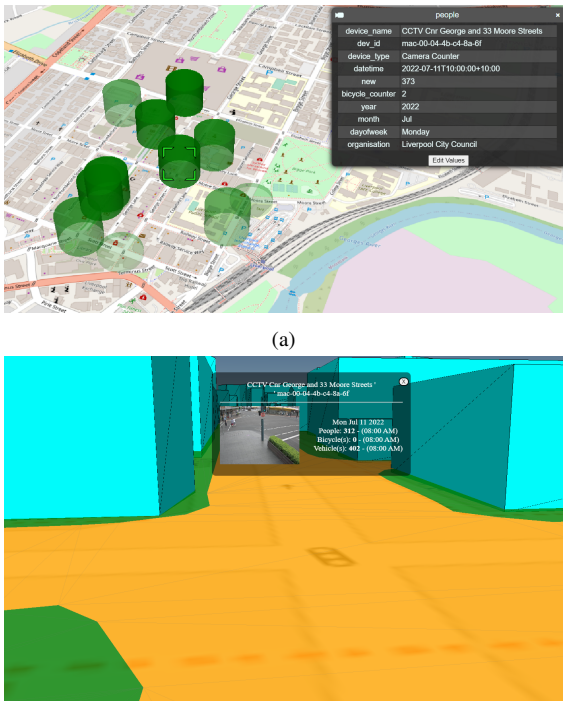
Figure 9. Unshaded areas (orange) vs. Tree coverage (green).

of few counting devices running at the time of the visualisation. The user can select the type of counting device to visualise (people, bicycle, vehicle or all of them simultaneously) and the latest feeds available will be queried and the scene updated accordingly. Options to hide or display the feed are also proposed, as well as the possibility to enable/disable the live updating (which occurs every 10 min, according to the pace of the sensors). We have chosen to use cylinders of the same radius since the sensors' roles are localised and provide only 1-dimensional value. The colour opacity and height of the cylinder are used to express the quantity counted (more opaque colours and higher means more counted entities). . Finally, the records obtained from the API queries are collected and displayed as attributes of the visualised feeds (see Fig. 10). Furthermore, to leverage the 3D capabilities of the DT, we also implemented a simulated camera view of selected CCTV to offer a contextual view of the pedestrian, bicycle and vehicle flows at a street level (Fig. 10(b)). Grounds (green) and road polygons (orange) are thereby coloured respectively according to the received pedestrian and vehicle feeds. A real picture from the simulated CCTVs along with corresponding feed values is also provided for reference.

6. CONCLUSION

This project demonstrates that a functioning digital twin can be built using already existing data and open-source technologies that can not only represent the contemporary environment across a range of detail, but also relay IoT data feeds directly via API, approaching a real-time representation. This information is visualised as a 3D immersive environment via accessible web-based tools, so no great technical hurdles or specialised training need be anticipated for a lay-user once the system is deployed. Notably, the formats employed in this demonstration are congruent with W3C and OGC standards, such as XML-based CityGML, GeoJSON and REST API. Having constructed this extensible foundation using web-enabled open-source resources, the potential for access, intuitive operation, sharing and update is maximised, while extended facilities are also available for implementation, such as facilitating secure/authorised access.

During the data procurement process, however, acquiring non-open data was difficult, even with formal data sharing agreements. For instance, any existing BIM data was usually kept with 3rd parties and unavailable. Similarly, pedestrian counters



(a)

(b)

Figure 10. (a) Visualisation of the feed from the people (green) counting devices. (b) Simulated camera view of a selected CCTV device with the road and pavement coloured according to people (green) and vehicle (orange) feeds.

on video surveillance cameras designedly reduced input data to a simple number/count decimated from the original video data, leaving further analysis into pedestrian direction, weather conditions or crowd behaviours unfeasible. In the future, interoperability with proprietary models will require multipartite agreement on, and compliance with, exchange standards to best ensure downstream value add when public data is feeding commercial implementations.

Efficiency, accuracy and detail need to be considered in order to create a convincing and trustworthy analogue of the original environment that is as easy to use as possible. In terms of scalability, increasing levels of detail must not only be accommodated, but a DT needs to work with incomplete data inputs. The ability to develop more complex datasets from known source data such as street and rail centerlines, building footprints and derived pedestrian walkways is an efficient way to employ proxy data if needed, and presents to opportunity to ingest higher quality data as they become available.

Using the use-case scenario of shadow casting as an example, 3D terrain is important for impactful 3D analysis like modelling shadow-fall from one building to another, the coverage of tree canopy shading, and, when combined with dynamic pedestrian data, the interactions between several factors taking into account slope, shade, time of day, season and the general built environmental context. The automated approach demonstrated in deriving integrated 3D environments plays a critical role in supporting higher quality analysis in subsequent phases of user-defined w orkflows, es pecially wh en su ch us ers ma y be non-expert in 3D geospatial applications.

An important feature of this DT is the ability to ingest many diverse types of urban data into a coherent environment and in

turn, output a variety of 'views' appropriate for the aims of the end-users. The next phase of this project will include user testing with both expert and non-expert users to elicit user feedback regarding ease of use and suggestions for future development.

Finally, regardless of the purpose of a DT, it is important that organisations producing or collecting data ensure their durability as much as possible. While this is a significant research topic on its own, we believe that open standards are the right way to go for future-proofing today's data for the DTs of tomorrow. They also remain the current best practice in terms of interoperability through avoidance of unharmonized data schemas and commercial/proprietary lock-ins, while remaining compatible with new technological advances (e.g., API, web support, etc.).

ACKNOWLEDGEMENTS

This project has been funded by FrontierSI, the Australian Urban Research Infrastructure Network (AURIN) and UNSW Research Infrastructure.

REFERENCES

Akahoshi, K., Ishimaru, N., Kurokawa, C., Tanaka, Y., Oishi, T., Kutzner, T., Kolbe, T. H., 2020. i-URBAN revitalization: conceptual modeling, implementation, and visualization to-wards sustainable urban planning using CityGML. *ISPRS An-nals of the Photogrammetry, Remote Sensing and Spatial In-formation Sciences*, 4, 179–186.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., Çöltekin, A., 2015. Applications of 3D city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4), 2842–2889.

Billen, R., A-F. Cutting-Decelle, C. Métral, G. Falquet, S. Zlatanova, and O. Marina, 2015, Challenges of Semantic 3D City Models: A Contribution of the COST Research Action TU0801, *3D Printing: Breakthroughs in Research and Practice*, Chapter 16, pp. 296 – 305

BITRE, 2022. Trainline 7. Bureau of Infrastructure and Transport Research Economics Publications. bitre.gov.au/publications/2019/trainline-7 (03 May 2022).

Boje, C., Guerriero, A., Kubicki, S., Rezgui, Y., 2020. Towards a semantic Construction Digital Twin: Directions for future research. *Automation in Construction*, 114, 103179.

Cesium GS, I., 2022. CesiumJS. 3D geospatial visualization for the web. cesium.com/platform/cesiumjs (26 April 2022).

Dimitrov, H., Petrova-Antonova, D., 2021. 3d city model as a first step towards digital twin of Sofia city. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 43, 23–30.

Emgård, L., S. Zlatanova, 2007, Design of an integrated 3D information model, *Urban and Regional Data Management: UDMS Annual 2007*, CRC Press, pp. 143-156

Förstner, W., 1999. 3D-city models: Automatic and semiautomatic acquisition methods.

Francisco, A., Mohammadi, N., Taylor, J. E., 2020. Smart city digital twin–enabled energy management: Toward real-time urban building energy benchmarking. *Journal of Management in Engineering*, 36(2), 04019045.

Gobeawan, L., Lin, E., Tandon, A., Yee, A., Khoo, V., Teo, S., Yi, S., Lim, C., Wong, S., Wise, D. et al., 2018. Modeling trees for virtual Singapore: from data acquisition to CityGML models. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences,* 42. 55-62

Grieves, M., 2014. Digital twin: manufacturing excellence

Hsu, L. S., Obe, R., 2021. PostGIS in action. Simon and Schuster.

Ivanov, S., Nikolskaya, K., Radchenko, G., Sokolinsky, L., Zymbler, M., 2020. Digital twin of city: Concept overview. *2020 Global Smart Industry Conference (GloSIC)*, IEEE, 178–186.

Kolbe, T. H., Gröger, G., Plümer, L., 2005. Citygml: Interoperable access to 3d city models. *Geo-information for disaster management*, Springer, 883–899.

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 Sci-ence*, 88(1), 43–61.

QGIS Development Team, 2022. QGIS Geographic Information System. QGIS Association.

Ruohomäki, T., Airaksinen, E., Huuska, P., Kesäniemi, O., Martikka, M., Suomisto, J., 2018. Smart city platform enabling digital twin. *2018 International Conference on Intelligent Systems (IS)*, IEEE, 155–161.

Schrotter, G., Hürzeler, C., 2020. The digital twin of the City of Zurich for urban planning. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 88(1), 99–112.

Sinnott, R. O., Bayliss, C., Bromage, A., Galang, G., Grazioli, G., Greenwood, P., Macaulay, A., Morandini, L., Nogoorani, G., Nino-Ruiz, M. et al., 2015. The Australia urban research gateway. *Concurrency and Computation: Practice and Experience*, 27(2), 358–375.

Tao, F., Zhang, H., Liu, A., Nee, A. Y., 2018. Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4), 2405–2415.

White, G., Zink, A., Codecá, L., Clarke, S., 2021. A digital twin smart city for citizen feedback. *Cities*, 110, 103064.

Wright, L., Davidson, S., 2020. How to tell the difference between a model and a digital twin. *Advanced Modeling and Simulation in Engineering Sciences*, 7(1), 1–13.

Yan, J., Diakité, A. A., Zlatanova, S., 2019a. A generic space definition framework to support seamless indoor/outdoor navigation systems. *Transactions in GIS*, 23(6), 1273–1295.

Yan, J., Zlatanova, S., Aleksandrov, M., Diakite, A. A., Pettit, C., 2019b. Integration of 3D Objects and Terrain for 3D Modelling Supporting the Digital Twin. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W8, 147–154. https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/IV-4-W8/147/2019/.

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T., Kolbe, T. H., 2018. 3DCityDB-a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospatial Data, Software and Standards*, 3(1), 1–26.