DEVELOPMENT OF A DIGITAL TWIN FOR THE MONITORING OF SMART CITIES USING OPEN-SOURCE SOFTWARE

R. R. Sta.Ana*, J. E. Escoto, D. Fargas Jr., K. Panlilio, M. Jerez, C. J. Sarmiento

Training Center for Applied Geodesy and Photogrammetry, University of the Philippines Diliman, Quezon City 1101, Philippines - rcstaana1@alum.up.edu.ph, jimescoto@outlook.com, (dcfargas, mvjerez, cssarmiento)@up.edu.ph, kppanlilio@gmail.com

KEY WORDS: Smart Cities, Digital Twin, Urbanization, 3D Visualization, WebGIS, Geographic Information Systems

ABSTRACT:

Cities are consistently motivated to come up with technology-driven solutions that aim to reduce the negative impacts of rapid urbanization. This paper explores open-source software as a platform in visualizing and developing a digital twin, which aids in mitigating the problem by running simulations and generating potential improvements through generated insights. The four essential components examined to develop the methodology are: (1) Visualization of Digital Model; (2) Identification of User Interface and Data Management Requirements; (3) User Interface Set-up and Configuration; and (4) Analysis and Simulations. Different tools for visualizing the city such as Unity3D, QGIS2threejs, and TerriaMap were explored and compared. Though Unity3D and QGIS2threejs can visualize 3D city models, TerriaMap was favored for its capability to visualize large areas in 3D and to create customizable user interfaces. User interface components were identified as well as handling and processing geospatial datasets. For the analysis and simulations, the Land Surface Temperature hotspot detection was performed and integrated into the system to demonstrate its potential to include other simulations in the future.

1. INTRODUCTION

The number of people moving to and settling in urban areas is steadily increasing. A UN projection shows that around half (50.1%) of the population in the Philippines will be residing in urban areas by the year 2028. One way of mitigating the negative impact of rapid urbanization and making it sustainable is the implementation of smart city solutions in managing the city. One such solution is the creation of a digital twin of the city where its status can be monitored (Johannes, 2019).

A digital twin can be used not only for monitoring the city, but also as a tool to aid in the planning of future developments. Simulations of future projects can be done within the digital twin to determine its effects on the city before it is implemented in situ (Hurtado and Gomez, 2021).

This paper aims to develop the methodology for creating a digital twin, using only open-source software. Using open-source software makes the methodology more accessible to local government units and other organizations especially in developing countries like the Philippines.

The digital twin will be developed for Iloilo City but the methodology should be applicable to other cities as well.

2. MATERIALS AND METHODS

Iloilo City was identified as the study site. The development of the digital twin consists of four (4) main parts: Visualization of Digital Model, Identification of User Interface and Data Management Requirements, User Interface Set-up and Configuration, and Analysis and Simulation.

2.1 Study Site

Iloilo City is the capital of the province of Iloilo. It is located in the Western Visayas region of the Philippines bounded by the Municipality of Oton to the west, Municipalities of Pavia and Leganes to the north, and by the Iloilo Strait to the south and the east. It has a land area of 78.34 sq. km and a population density of 5,719 per sq. km (Philippine Statistics Authority, 2015).

Iloilo City is classified as a highly urbanized city making it a prime pilot area for this study. The Philippines' Department of Information and Communications Technology (DICT) also echoed this sentiment, citing that the city has already met 6 of the 12 specific components under the Digital Cities Framework. (Lena, 2017).

Figure 1. Study Site - Iloilo City

* Corresponding author

2.2 Visualization of Digital Model

Available open-source 3D modelling software and its suitability for 3D visualization of GIS layers were explored: QGIS2threejs (Akagi, 2021), Unity3D downloaded from the Unity website (Unity Technologies, 2005), and TerriaMap (Terria, 2020).

Different factors were considered in visualizing the digital model of Iloilo City. The software must be capable of handling and managing large datasets. As the study will be making use of layers covering the whole Iloilo city (e.g., LIDAR DSM/DTM, building footprints, etc.), the software must be able to display these layers at an acceptable speed and the display must also be responsive to user input.

The software must also have the ability to display a user interface. These include displaying map legends and vector symbology, map generation and printing, as well as basic spatial tools (measure distance/ area).

Another thing to consider is the compatibility of usual GIS file formats (e.g. *.shp, *.tif, etc) with the software since we will be working mainly with GIS layers. This simplifies the workflow and minimizes the chance of errors from conversion of file formats.

2.3 Identification of User Interface and Data Management Requirements

An excellent option to create a digital twin comes along with a user-friendly interface with robust data management capabilities and control. These parts should be flexible and modular to adapt to certain needs of a city. It should have the ability to support different geospatial datasets and file types. These properties allow the digital twin to be sustainable in the long run. This section presents the core user-interface components that must be present in order to achieve that goal. The latter portion presents the basic process of configuring the webGIS, its data catalog, and basemap layers sections.

2.3.1 User Interface Components: The following components must be present to perform basic create, read, update, and delete (CRUD) operations for the digital twin of a smart city.

2.3.2 Data Catalog: This allows datasets to be searched, shared, and added to the map using this feature. It serves as a data management tool and an inventory of all the available datasets and their corresponding metadata.

2.3.3 Data Upload: This allows users to upload their own data to the map, provided that the data's file format is supported.

2.3.4 Workbench: This shows a list of activated data layers added to the map. This feature includes additional controls to toggle on/off the layers, to change opacity setting, to show some legends, etc.

2.3.5 Basemap Layers: This serves as a reference map that provides the necessary background setting that will put into context the succeeding data to be overlaid.

2.3.6 Map Display & Tools: The map display renders the georeferenced layers which enables the user to view, rotate, pan, zoom and interact with multiple datasets. These maps can also be shared and printed.

2.3.7 Attribute Information: Apart from visualization, another important component is the capability to integrate and display relevant attribute information such as semantic data about the 3D buildings or any supplementary information such as statistics, charts.

2.3.8 Data Layer Support: The interface must be capable of handling both vector and raster files. Common file types (*.shp, and *.geojson for vector, *.tif for raster) must be supported.

2.3.9 Related Maps: The interface must be able to show maps and other dashboards related to the management of the subject area. These are usually other initiatives by other agencies for other sectors of the local government. Nevertheless, the digital twin must be able to integrate this into the system or provide external links to the other initiatives.

2.4 User Interface Set-up and Configuration

The user interface set-up is based on a modified version of TerriaMap, a full-featured application built on TerriaJS. TerriaJS is a feature rich open source solution for building amazing spatial data federation web platforms (TerriaJS, 2014). Currently, it is being supported by NICTA and CSIRO Data61 in collaboration with the Australian Government (TerriaJS, 2014).

2.4.1 Web GIS Initialization: To deploy the WebGIS using TerriaMap, run the following commands in the VSCode terminal (make sure that node and git-bash are installed):

```
git clone https://github.com/TerriaJS/TerriaMap.git
cd TerriaMap
export NODE_OPTIONS=--max_old_space_size=4096
npm install && npm run gulp && npm start
```

These commands have been sourced from the TerriaJS V8 Documentation Guide (TerriaJS, 2021). By running these commands, the platform can be accessed locally using the browser and entering the following URL: `http://localhost:3001`. The server can be shut by using the `npm stop` command.

2.4.2 Data Catalog Configuration: The main dataset that is used for this study is the Geography Markup Language (GML) file which contains the 3D buildings inside Iloilo City. Other datasets include the barangay boundaries shapefile. TerriaMap's data catalog feature is defined in one of the 'initialization files' of the cloned TerriaMap folder, which in our case is found in wwwroot/init/simple.json (TerriaJS, 2021). The catalog file name file is then added in the config.json file (Figure 2) without the *.json extension.

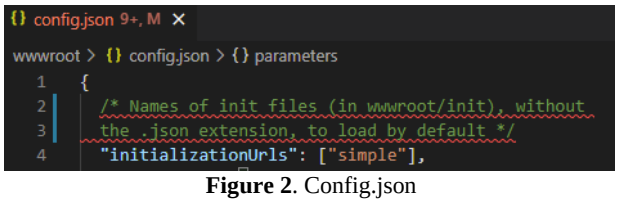

Figure 2. Config.json

This catalog feature supports broad types of data including the catalog group, catalog item, and catalog function - each of which must have its corresponding name and type (TerriaJS, 2021).

Figure 3 shows an example of how one can configure the data catalog. Here, we have a catalog group called "Administrative Boundaries" which includes two members namely "Barangays" and "Districts", both of which are of type geojson. These

geojson files are stored and can be accessed using the URL link provided. In this case, the files are located in the lungsod_files folder (Figure 4), a sibling folder of the "init" folder.

```
"catalog": [
 {
  "name": "Administrative Boundaries",
  "type": "group",
  "members": [
   {
    "name": "Barangays",
    "type": "geojson",
    "URL": "lungsod_files/barangays.geojson"
   },
   {
     "name": "Districts",
     "type": "geojson",
     "URL": "lungsod_files/roads.geojson"
    }
  ]
 },
]
```

Figure 3. Simple.json: catalog with local file paths example

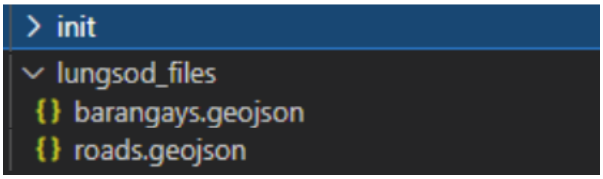

Figure 4. Sample local files

When data is stored online, however, one can change the URL parameter depending on the platform used, as seen in Figure 5.

```
"catalog": [
 {
  "name": "Administrative Boundaries",
  "type": "group",
  "members": [
   {
    "name": "Barangays",
    "type": "geojson",
    "URL": "https://lungsod.herokuapp.com
            /api/mediafiles/uploads/geojson/
            barangay_boundaries.geojson"
   },
   {
     "name": "Districts",
     "type": "geojson",
     "URL": "https://lungsod.herokuapp.com/
            api/mediafiles/uploads/geojson/
            1b_7Districts-wgs84.geojson"
    }
  ]
 }
```

Figure 5. Simple.json: online file paths example

More information about other catalog groups and items and their corresponding types and parameters are documented in TerriaMap's Documentation Guide (TerriaJS, 2021).

2.4.3 Basemap Layers Configuration: TerriaMap already contains four (4) default basemap layers. These are the Natural Earth II, NASA Black Marble, Positron (Light), Dark Matter (TerriaJS, 2021). However, additional basemap layers can be added in a.JSON folder. Again, each item must include a name, its proper type, and a URL to work, as shown in Figure 6.

```
"baseMaps": {
    "items": [
      {
        "item": {
          "id": "osm",
          "name": "OSM",
          "type": "open-street-map",
          "URL": "https://tile.openstreetmap.org"
        },
        "image": "images/basemaps/osm.png"
      },
      {
        "item": {
          "id": "bing-labels",
          "name": "Bing Maps with Labels",
          "type": "ion-imagery",
          "attribution": "Bing",
          "ionAssetId": 3,
          "ionAccessToken": "",
          "ionServer": "https://api.cesium.com"
        },
        "image": "images/basemaps/bing
                  -aerial-labels.png"
      }
    ],
  ]
}
```

Figure 6. Simple.json: basemap example

Note, however, that for items that are fetched from other sources such as cesium ion - a platform for 3D geospatial datasets - the corresponding parameters (asset id, access token, and server) must be specified.

The default basemap and the basemap used to preview data can be assigned by putting in the id of the desired basemaps, as displayed in Figure 7.

```
    "defaultBaseMapId": "bing",
    "previewBaseMapId": "osm"
```

Figure 7. Simple.json: basemap default

2.5 Analysis and Simulations: Another component of the digital twin is data analysis. The study aims to develop a software or a workflow using only open-source software for performing data analysis which can be integrated into the digital twin. In this study, this will be a hotspot detection workflow.

The workflow will make use of Landsat 8 satellite images to obtain land surface temperature data which is necessary for the analysis. Landsat 8 was selected due to it being free, and also easily accessible through a Google Earth Engine (GEE) web app developed by Project GUHeat, a previous DOST-funded project (Project GUHeat, n.d.). This GEE web app under Project GUHeat will also be used for this study.

The methodology for hotspot detection used in this study is adopted from a paper by Project GUHeat (Baloloy et. al, 2019). The methodology was adjusted to cover images from a shorter time frame as only the hotspot detection component was being done. In Baloloy et. Al, the analysis was done on yearly Landsat images from 1989-2019. In this study, only the available Landsat 8 images available for the year 2020 were used. The images were obtained using the GEE webapp and clipped to include only Iloilo city. A total of 21 images were processed and downloaded, however only 10 were used in the analysis due to cloud cover. Images with more than 40% cloud cover were excluded. This threshold was selected through manual inspection of the cloud cover percentage of each image. The allowable cloud cover must be low enough to minimize the cloudy pixels in the analysis but high enough to not exclude most of the available datasets.

The hotspot detection was done by first dividing the images into groups of 3 with each group having temporally consecutive

images. The temperature values in each image were then split into 7 classes (-3 to +3) from hottest to coolest with -3 being the coolest and +3 being the hottest. Similar to Baloloy et. al, the breaks between classes were determined using the Jenks natural breaks method but was done through Python instead of ArcGIS, a commercial software. The images in each group were then summed up to determine whether the area was a hotspot. In order for a pixel to be classified as a hotspot, it needs to be in the hottest class (+3) for at least 50% of the number of images and only have values in the top 2 hottest classes (+2 and +3). In this case with 3 images per group, only pixels with sums 8-9 are considered hotspots. This method is only applicable to pixels with a value in all 3 images, which is one of the main reasons why the cloud cover threshold was implemented. All the raster computations were done using Python but an alternative, which is also free and without much coding, is through QGIS. The output was then exported into a GeoJSON format to make it easy to import into the digital twin. The complete workflow can be seen in Figure 8.

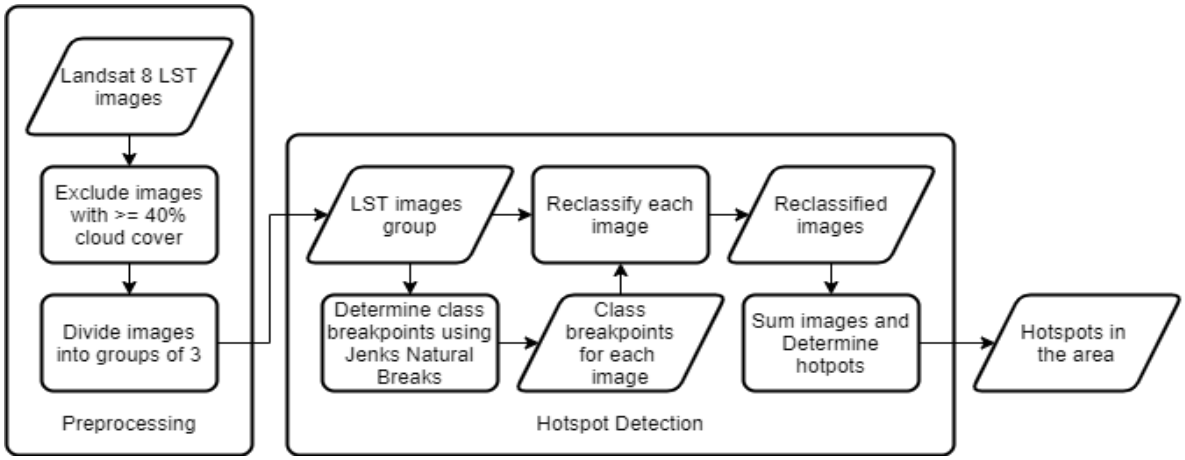

Figure 8. Hotspot detection workflow

3. RESULTS AND DISCUSSION

3.1 Comparison of Different Visualization Platforms

Figures 9 to 11 show the different visualization outputs from the different 3D modelling software used. The options explored for 3D visualization were Unity3D, a game development software; TerriaMap, a Javascript library combined with CesiumJS; and Qgis2threejs, a plugin for QGIS. All of these are capable of displaying 3D models and providing the user with the means to interact with said model. Unity3D has the advantage of being able to package the 3D visualization into a standalone software, while TerriaMap is more suitable for developing webapps. On the other hand, Qgis2threejs, being a QGIS plugin, supports a wider range of GIS file formats natively.

Currently, the 3D visualizations can display terrain, natural and man-made features (e.g. trees, buildings, roads), and other data layers as 3D models. Other data display modes, such as plots and graphs, will be added in future iterations along with more control options for the user.

Figure 9. 3D visualization output of Iloilo City using Qgis2threejs

Figure 10. 3D visualization output of Iloilo City using Unity3D with Blender

Figure 11. 3D visualization output of Iloilo City using TerriaMap

3.2 User Interface and Data Management

The TerriaMap comes with a simple and user-friendly interface that is divided into five sections, as referenced in TerriaMap Figure 11.

The left panel (Section A) of the TerriaMap application contains the following features:

Feature	Description
Search for locations	Allows searching for specific places in the world map, or in the data catalog (Figure 12). Zooms into the selected place.
Explore map data	Opens data catalog to browse and load specific datasets.
Upload	Enables users to view datasets locally in the map, either through a local source or a web data source. Does not save uploaded files.
Workbench	Layer items appear in the workbench when loaded in the map. Different controls include visibility, zoom-in, and remove. Metadata can also be viewed.

Table 1. Features found in Section A

The right side contains the controls (Section B), which has the following features:

Feature	Description
Gyroscope control	• Outer ring rotates the map view and contains the compass. • Inner ring tilts and rotates the map view, simultaneously.
Zoom control	Used to zoom in, zoom out or reset to default zoom level on the map.
Location	Plots current GPS location of user in map when given access
Compare	Allows side-by-side comparison of map terrain datasets
Pedestrian mode	Enables users to access the map in a walking mode
Line measure	Measures distances between points on the map

Table 2. Features found in Section B

The about and related maps (Section C) shows more about the project and other related datasets:

Feature	Description
About	When clicked, users will be redirected to a section that contains the project description.
Related Maps	When clicked, users will be redirected to a section that features other projects or maps that are relevant to the project.

Table 3. Features found in Section C

The upper-right side contains more map controls (Section D):

Feature	Description
Map Settings	Used to customize map settings such as map view, show terrain on compare feature, basemaps, and image quality. Timelines and image optimization options can be toggled.
Share/ Print	Allows sharing and printing of the current map view.
Story	Allows contextual information to be added to a dataset.

Table 4. Features found in Section D

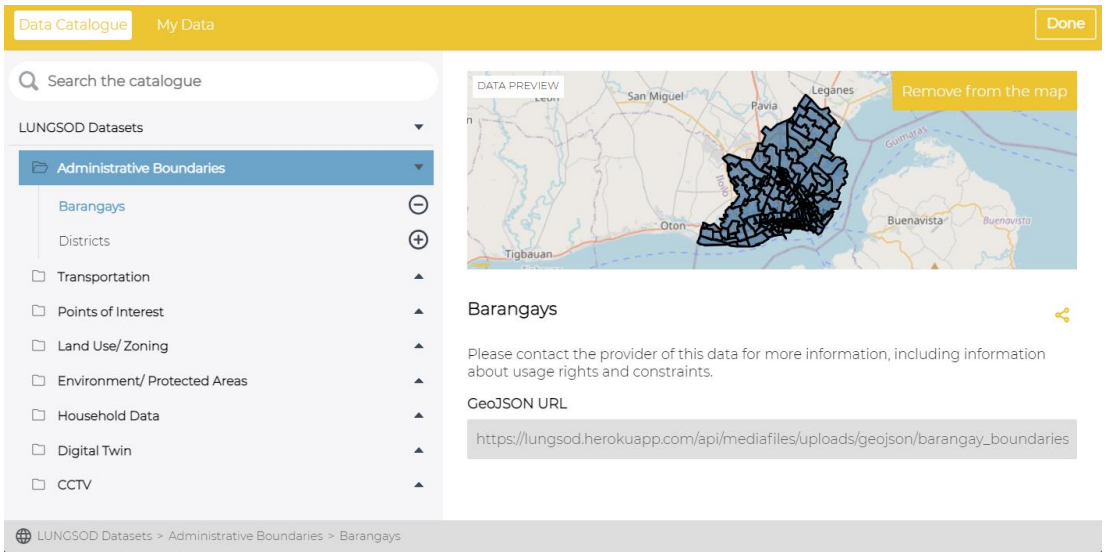

Figure 12. TerriaMap Data Catalog

Shown in Figure 12 is the result of configuring the data catalog. The left-hand portion contains all the available datasets, a control to search for desired data. The right-hand portion, on the other hand, showcases a preview of the layer to be added and other information about it.

Configuring the basemap will result in something similar to the figure below. These basemap layers can be accessed in the map settings control (included in Section D).

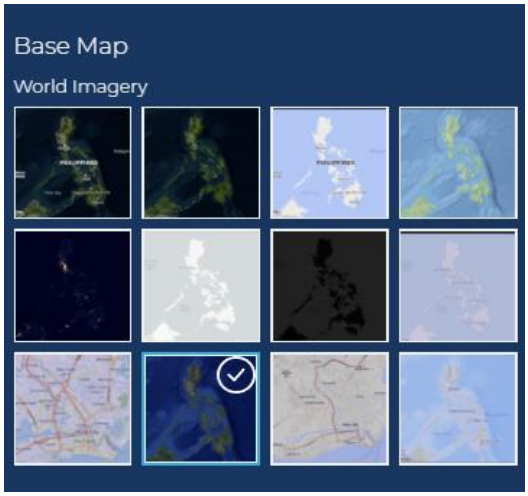

Figure 13. Basemap layers

The map display (Section E) renders all the activated layers including the basemap. It contains the feature information display that pops up when a feature is selected from a dataset (in this case, a building feature from the 3D buildings dataset).

3.3 Data Analysis

Some of the results of the data analysis are shown in Figure 14**.** Almost all of the hotspots detected were located in built-up/bare soil areas. This was expected as the analysis used daytime images only and vegetated areas heat up slower during the day. There were however some vegetated areas which were detected as hotspots. This may be attributed to the fact that even with the cloud cover thresholding applied, large areas in the image sets are still unusable which may have led to lower temperature ranges being considered as hotspots. Future studies should still be done to refine the analysis as currently it is at a level that may only be useful for citizens in planning their day-to-day activities as its reliability is dependent on the available clear datasets. A cloud-filling algorithm may be employed to maximize the number of usable pixels. This however, might lead to a lower accuracy output depending on the accuracy of the cloud-filling algorithm used. A longer time frame may also be used to increase the number of usable images. Validation using in-situ temperature measurements is also needed.

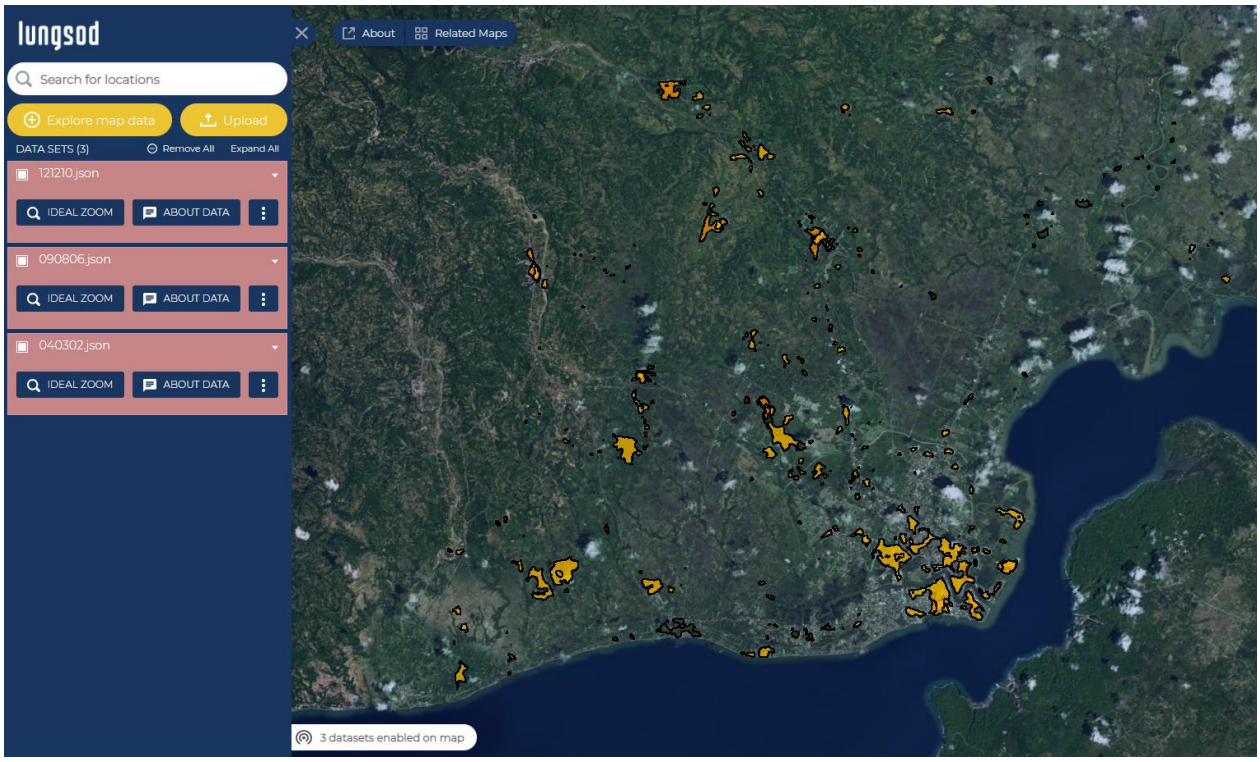

Figure 14. Sample Hotspot Detection Output

4. CONCLUSIONS AND RECOMMENDATIONS

The study was able to develop a framework for creating a digital twin using only open-source software. Various visualization software were explored and after considering several factors (e.g. compatibility with GIS datasets, level of customization possible, etc), TerriaMap was selected as most suitable for the visualization component of this framework. TerriaMap also already includes a user-friendly interface and built-in tools commonly used in GIS applications (e.g. location search, GIS data upload, etc.) This minimizes the need for additional feature development. For data storage, the datasets will be hosted on a private server. These will be accessible to the users through the internet using the digital twin software. This lessens the storage and processing load on the client, making it accessible to a wider range of users. The option to upload and display local datasets into the digital twin is also available for the users. For data analysis, a hotspot detection workflow was developed using only open-source software. The outputs from this workflow can then be easily accessed through the digital twin.

ACKNOWLEDGEMENTS

This research was done as part of the A Link-Up of Geomatics and Social Science Research for the Development of Smarter Cities (LUNGSOD) Project. The Project was implemented by the University of the Philippines Training Center for Applied Geodesy and Photogrammetry (TCAGP), through the support of the Department of Science and Technology (DOST) of the Republic of the Philippines and the Philippine Council for Industry, Energy, and Emerging Technology Research and Development (PCIEERD).

REFERENCES:

Akagi, M. (2021). Minorua/qgis2threejs: [QGIS plugin] a 3D scene web exporter. GitHub. Retrieved September 1, 2021, from https://github.com/minorua/Qgis2threejs.

Baloloy, A., Sta. Ana, R. R., Cruz, J. A., Blanco, A. C., Lubrica, N. V., Valdez, C. J., & Cajucom, E. P. (2019). Spatiotemporal multi-satellite biophysical data analysis of the effect of urbanization on land surface and air temperature in Baguio City, Philippines. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-4/W19, 47–54. https://doi.org/10.5194/isprs-archives-xlii-4-w19-47-2019

Hurtado, P. (2021). Smart City Digital Twins are a new, hi-tech tool for scenario planning. American Planning Association. Retrieved October 18, 2021, from https://www.planning.org/planning/2021/spring/smart-city-digital-twins-are-a-new-tool-for-scenario-planning/.

Johannes, M. (2020). Smart cities: The strategy to face the urbanization challenges. Engineers & Architects. Retrieved October 18, 2021, from https://www.e-zigurat.com/blog/en/smart-cities-urbanization-challenges/.

Lena, P. 2017. DICT names Iloilo as potential digital city. https://www.pna.gov.ph/articles/1016766

Philippine Statistics Authority (2017). 2015 Census of Population, Report No. 3 – Population, Land Area, and Population Density

Project GUHeat. (n.d.). LST Landsat v2. GUHeat Engine. Retrieved October 11, 2021, from https://projectguheat.users.earthengine.app/view/lst-landsat-v2

Terria. (2020). TerriaMap, Github repository. Retrieved September 1, 2021, from https://github.com/TerriaJS/TerriaMap.

TerriaJS. (2014). TerriaJS. Retrieved October 12, 2021, from https://terria.io/.

TerriaJS. (2021). TerriaJS V8 Documentation Guide. TerriaJS. Retrieved September 18, 2021, from https://docs.terria.io/guide/.

Unity Technologies (2005), Unity [Computer software]. Retrieved September 18, 2021, from http://unity.com